1. **Using the data synthesis R script provided by the instructor as part of the week 11 assignment instructions, produce datasets of the following sizes, and fit deep learning models with the configurations shown below. Associated with each model, record the following performance characteristics: training error, validation (i.e., holdout set) error, time of execution. Use an appropriate activation function.**

Datasize	**Configuration**	**Training error**	**Validation/test error**	**Time of execution**
1000	1 hidden layer 4 nodes	0.5531	0.7136	8.06s
10000	1 hidden layer 4 nodes	0.3269	0.3206	5.72s
100000	1 hidden layer 4 nodes	0.0696	0.0810	51.89s
1000	2 hidden layers of 4 nodes each	0.6049	0.6080	2.73s
10000	2 hidden layers of 4 nodes each	0.5380	0.5266	7.87s
100000	2 hidden layers of 4 nodes each	0.0758	0.0830	56.26s

2. **Based on the results, which model do you consider superior, among the deep learning models?**
   Best model: The 100000 data size with 2 hidden layers of 4 nodes each, achieved the lowest training and validation error, which indicates the best overall performance among all the neural networks that are tested. Among the deep learning models having 1 hidden layer with 4 nodes with various data sizes, 100k size performed well with slightly better than other 2 sizes. Whereas among the deep learning models having 2 hidden layers with 4 nodes each with various data sizes, the one with 100k size outperformed the remaining with the least validation error of 0.8030. If the execution time is a critical concern,we can slightly favor the 1-layer model on 100000 rows. But for model accuracy and learning, the 2-layer model is superior. So, overall performance and accuracy wise, the 2-layer model outperformed the remaining models.
3. **Next, report the results (for the particular numbers of observations) from applying xgboost (week 11 – provide the relevant results here in a table). Comparing the results from XGBoost and deep learning models, which model would you say is superior to others? What is the basis for your judgment?**

Model used	Dataset size	Configuration	Testing-set predictive performance	Training error	Validation /test error	Time taken for the model to be fit
XGBoost in Python via scikit-learn and 5-fold CV	1000	-	0.9520	-	-	0.30s
	10000	-	0.9753	-	-	0.51s
	100000	-	0.9869	-	-	1.46s
XGBoost in R – direct use of xgboost() with simple cross-validation	1000	-	0.9350	-	-	0.28s
	10000	-	0.9691	-	-	0.66s
	100000	-	0.9790	-	-	1.79s
XGBoost in R – via caret, with 5-fold CV simple cross-validation	1000	-	0.9320	-	-	1.38s
	10000	-	0.9394	-	-	1.39s
	100000	-	0.9436	-	-	8.59s
Deep learning/ Neural network	1000	1 hidden layer 4 nodes	-	0.5531	0.7136	8.06s
	10000	1 hidden layer 4 nodes	-	0.3269	0.3206	5.72s
	100000	1 hidden layer 4 nodes	-	0.0696	0.0810	51.89s
Deep Learning/ Neural network	1000	2 hidden layers of 4 nodes each	-	0.6049	0.6080	2.73s
	10000	2 hidden layers of 4 nodes each	-	0.5380	0.5266	7.87s
	100000	2 hidden layers of 4 nodes each	-	0.0758	0.0830	56.26s

After analyzing the performance of XGBoost models (in Python and R) and deep learning models (with 1 and 2 hidden layers), we can determine the superior model based on three key criteria: predictive performance, training time, and scalability.

Predictive Performance (Test Accuracy or Validation Error): XGBoost (Python) achieved the highest test accuracy across all dataset sizes. In contrast, deep learning models reported validation errors ranging from 0.7136 (worst) to 0.0810 (best). Even the best neural network (100k rows, 2 hidden layers) has higher error than XGBoost's accuracy on just 1000 rows.

Training Time: XGBoost (Python) model with 100k rows trained faster: 1.46s vs. 56.26s for the best deep learning model. Deep learning models require significantly more time, especially on larger datasets.

Scalability: XGBoost both in R and Python shows consistently strong results across various dataset sizes. Neural networks struggle at small sizes (high error > 0.5) and only improve significantly at 100k, yet still don't outperform XGBoost.

- XGBoost (Python implementation using scikit-learn and 5-fold CV) shows stable and improving performance with increasing data.
- Neural networks show poor performance on small datasets.
- This model showed better generalization, evident from its high performance even without overfitting, whereas deep learning models have higher training errors especially on small data.

Conclusion: So to conclude, XGBoost (Python implementation using scikit-learn and 5-fold CV) is the superior model across all dataset sizes.